Blockchain Forks: A Formal Classification Framework and Persistency Analysis

Schär, Fabian

Center for Innovative Finance, University of Basel, Faculty of Business and Economics, University of Basel

2020

Blockchain Forks: A Formal Classification Framework and Persistency Analysis

Fabian Schär[1]

Center for Innovative Finance
Faculty of Business and Economics
University of Basel

Abstract: Blockchain forks can have severe economic implications, sow uncertainty and undermine trust. In this paper, we introduce a formal framework to study the emergence, persistency and economic consequences of blockchain forks. We argue that blockchain forks can be process- or protocol-based and emerge unintentionally or deliberately. We then proceed with a sub-classification of protocol-based forks and study the circumstances under which a chain split may become permanent. It can be shown that the persistency of a fork depends on the nature of the change to the consensus rules and on the relative allocation of the consensus-relevant resources. Lastly, we discuss business implications as well as potential consequences for policy makers and practitioners.

JEL Classification: C38, G15, G23

Keywords: blockchain, chain split, cryptoasset, distributed ledger, fork.

[1]Address: Universität Basel, Peter Merian-Weg 6, CH-4052 Basel, Switzerland.
Phone: +41 61 207 33 25
Email: f.schaer@unibas.ch

1 Introduction

For economists, cryptocurrencies are very difficult to analyze. On the one hand, we would like to assume that the rich history of economic research provides us with the tools necessary to study this relatively new phenomenon. After all, our models are general and should be applicable to any asset and monetary regime. On the other hand, there are some characteristics in which cryptocurrencies differ fundamentally from everything we have seen so far. The concept of a truly decentralized and permissionless digital asset raises new questions in the context of monetary policy, transaction finality and governance. One of the unique characteristics, which is a direct consequence of decentralization, are blockchain forks, i.e. potentially persistent disagreements on the current state of the ledger that may cause the blockchain to split and create two or more competing database instances.

Forks have been discussed as early as the original Bitcoin paper (Nakamoto, 2008). They have been identified as a potential problem in security surveys (Lin and Liao, 2017) and analyzed in terms of their compatibility (Berentsen and Schär, 2017; Zamyatin et al., 2018). Additionally, there have been case study analyses for specific forks (Islam et al., 2019; Kiffer et al., 2017).

Although most economists would likely agree that chain splits are an interesting topic, there is surprisingly little economic research on forks. In fact, there is not even a clear definition of the word *fork*. Instead, the term is used to describe a variety of situations. Sometimes it refers to a temporary state of uncertainty, when two or more competing blocks are created at approximately the same time; at other times it refers to a change of the consensus rules that may or may not cause the network

to split and a competing version of the Blockchain to emerge.

Despite the lack of a clear definition, forks are very real and can have devastating economic effects. They cause confusion and mix-ups when there are several competing versions of a cryptoasset. Moreover, they may open up new attack vectors through the potential of a replay attack and jeopardize data protection through cross-chain identity analyses. Tax and legal questions (Himmer et al., 2018; Landoni and Pieters, 2019; Webb, 2018), as well as the need for large financial investments to ensure the compatibility of storage solutions with new forks, introduce additional inefficiencies and, when the forked blockchain is used for the tokenization of off-chain assets or serves as a platform for interoperable decentralized applications, the situation gets even more complicated. To sum up, forks undermine the trust in the system and sow uncertainty.

There is no lack of examples for Blockchain forks. As of May 2020, there are more than forty listed cryptoassets that originated from a Bitcoin chain split and five that split from Ethereum.[2] While most forks are economically irrelevant, the two most prominent examples highlight the uncertainty involved in a forking event (Berentsen and Schär, 2018). On 20 July 2016, a large portion of the Ethereum community decided to change the consensus rules in response to the DAO hack (Mehar et al., 2017). Some members of the community wanted to perform an irregular state change and thereby undo the hack, while others decided to stick to the original rules and the blockchain's immutability. The resulting fork led to two competing chains. Ethereum Classic (following the original rules) and Ethereum (where the hack was undone).

[2]The website `https://forks.net` provides an extensive list of first order Bitcoin and Ethereum forks. Accessed 25 May, 2020.

On 1 August 2017, a long standing disagreement in the Bitcoin community led to the Bitcoin Cash fork. The difference was mainly in block size. While the Bitcoin community kept a limit of 1 MB blocks and was in favor of second layer scaling solutions, Bitcoin Cash followers wanted to scale on-chain and implemented a 8 MB block size.

Interestingly, the two examples had a completely different outcome. In the case of Bitcoin, the blockchain with the modified rules corresponds to approximately 2.5% of the Bitcoin market cap. With Ethereum, it is the other way around. The DAO fork version became the dominant chain and has a market cap that is approx. 30 times higher than the market cap of the blockchain with the original consensus rules.[3]

In addition to forks that materialize in chain splits, there are certain types of forks that dissolve immediately. Bitcoin's Segregated Witness (SegWit) modification changed some of the rules. However, as a result of the way the change was implemented and the allocation of consensus-relevant resources, the SegWit version was uncontested.

Considering these examples, the diversity of potential outcomes and the corresponding economic consequences, it is essential to have a good understanding of blockchain forks. It is particularly important to be able to distinguish between the different types of forks. That is why this article introduces a formal classification framework, in an attempt to untangle the various meanings of the term *fork* and to provide a foundation for further research.

After this short introduction, we propose our framework in Section 2. We differen-

[3]Market cap data source: `coinmarketcap.com`. Data obtained 25 May, 2020.

tiate between process-based and protocol-based forks, study different sub-categories of protocol-based forks and argue that our classification allows us to explain the severity and long-term implications. In Section 3, we study the persistency of forks and demonstrate how the probability that a permanent protocol-based fork emerges depends on two factors: the allocation of consensus-relevant resources and the relative size of the intersection of the two block acceptance sets. In Section 4, we briefly discuss our results and conclude with areas that need further research.

2 Formal Classification Framework

Let us assume that there are two consensus-relevant nodes (CRN) M_A and M_B. Both are validating and relaying transactions and are trying to extend the blockchain by assembling candidate blocks. Candidate blocks are transaction collections that compete for the inclusion into the blockchain. They will get accepted if they are created in accordance with the blockchain's consensus rules. In particular, the candidate block and all included transactions must be cryptographically sound, exclusively reference valid outputs and provide solutions to the locking scripts (utxo model), result in valid state changes (account-based model) and satisfy the threshold criterion that artificially decreases the block creation speed.

Let us further assume that there is a finite block acceptance set S with $S \neq \emptyset$, that incorporates all of these criteria, so that a candidate block b will get accepted by any network participant who is employing S if a proposal $b \in S$ and refused if $b \notin S$. For the sake of readability, we denote confirmed blocks with b_h, where index

$h \in \{0, ..., N-1\}$ is a sequence number representing the block's position in the blockchain, i.e. the block height and N is the length of the dynamic block sequence vector. Figure 1 shows a simple block sequence (or blockchain) consisting of 4 blocks.

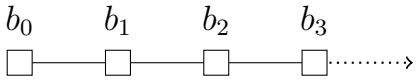


Figure 1: A simple block sequence

The decentralized nature of block creation can lead to disagreements, which may result in the emergence of two or more incompatible versions of the blockchain. This is usually referred to as a fork or chain-split. More precisely, we define a fork as a network state in which there are two (or more) active branches, with a common origin, i.e. a block from which these branches split. A generic example of this situation is shown in Figure 2, where the two branches with blocks b_3 and b_{3a} respectively, both originate from b_2.

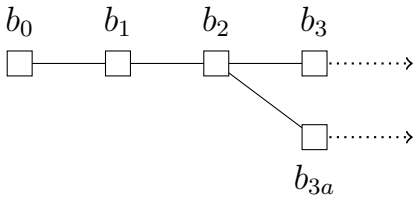


Figure 2: A simple fork

Some forks will only have a temporary effect and dissolve relatively quickly. Others may permanently split the network into two (or more) competing versions. Once we understand that forks differ significantly – not only in their type but also in the potential consequences – it becomes apparent that there is a need for a formal

analysis and a framework to assess the type and potential consequences of a fork.

Let us first differentiate between *process-based* forks and *protocol-based* forks. To do so, we must allow each CRN to choose its own block acceptance set. Let us denote the block acceptance set of CRN M_A with A and the block acceptance set of CRN M_B with B.

We call a fork process-based when it emerges, despite the two CRN M_A and M_B generating blocks in accordance with the same block acceptance set S, such that $A = B = S$. Since there are no differences in the rule set, the state will be determined exclusively by the allocation of consensus-relevant resources; hence, the name process-based.

We call a fork protocol-based when it emerges as a consequence of differences in the block acceptance sets, i.e. $A \neq B$. It may be caused due to differences in the interpretation or a change of the consensus rules. If, for example, M_A has a different understanding of the consensus protocol than M_B, and therefore does not accept M_B's newly created block, this may cause a protocol-based fork.

Process- and protocol-based forks can both emerge unintentionally or deliberately. Unintentionally means that the fork just happens without any party actively trying to create a fork. Deliberately means that at least one party wanted the fork to emerge and has taken steps to actively promote it. Table 1 summarizes our preliminary findings. The four categories will be described in the following sections.

	Process-based ($A = B = S$)	**Protocol-based** ($A \neq B$)
Unintentional	Probabilistic Block Race	Client Incompatibility • Soft Fork • Hard Fork • Forced Fork
Deliberate	Block Withholding & Forced Block Race	Rule Change • Soft Fork • Hard Fork • Forced Fork

Table 1: The 4 types of forks.

2.1 Process-based

An *unintentional* process-based fork is usually referred to as probabilistic block race. It emerges whenever two or more CRN find a new block at approximately the same time. This is possible due to the probabilistic nature of the block creation process, inherent to many consensus protocols. Let us turn to the example in Figure 2 and assume that, b_3 and b_{3a} have been created nearly at the same time. Both blocks have the same block height and therefore are conflicting, meaning that only one of these blocks can prevail. A probabilistic block race is a temporary phenomenon that resolves after a short period of uncertainty – in particular, when one of the competing blockchain versions gets extended by another block. Considering the *longest chain* consensus rule, or more precisely the highest accumulated difficulty (Nakamoto, 2008) or any of its variations (Sompolinsky and Zohar, 2015), the extension creates an incentive for all neutral CRN to join the dominant version.

Deliberate process-based forks are somewhat more complex and harder to resolve. They may originate from two distinct sources.

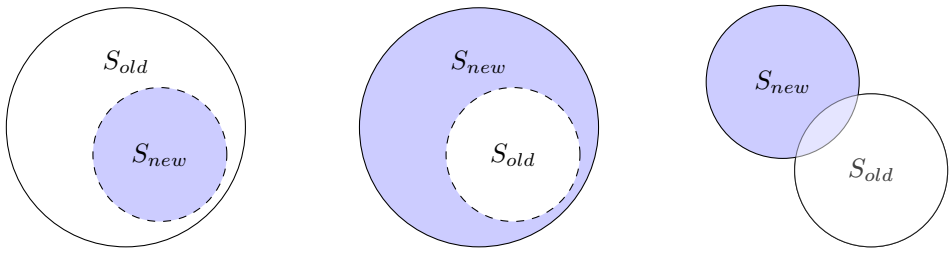
First, a CRN may decide to attack the dominant version of the blockchain by creating a new block that builds on an already confirmed block with $h(b_a) < N - 1$. If conducted successfully, this attack may render any blocks with $h \geq h(b_a)$ invalid. Attacks of this kind can only create temporary forks, which dissolve when the attacker is successful or decides to abort the attack.

Second, deliberate process-based forks may also emerge, when a CRN succeeds in creating a new block but decides to withhold the information regarding the existence of this block. The CRN may use the newly found block as a parent and immediately start working on the next block without telling anyone that it has found a valid block - giving it a head start and potentially letting others partially waste their consensus-relevant resources. This phenomenon is usually referred to as selfish mining or block withholding (Eyal and Sirer, 2018; Sapirshtein et al., 2016). The practical implications of these forks are debatable and there are certain defense mechanisms (Zhang and Preneel, 2017). While block withholding may create severe uncertainty in the short run and potentially undermine the perceived immutability characteristics of the blockchain, it is important to note that (a) the CRN will only be able to do this successfully if it is in control of a large portion of the network's consensus-relevant resources and (b) that this type of fork is a temporary phenomenon that will automatically dissolve once the CRN discloses the information. Nonetheless, forks of this kind may potentially lead to transactions being reordered or completely removed from the dominant version of the blockchain.

2.2 Protocol-based

An *unintentional* protocol-based fork may emerge in the case of a bug in the software client. Let us assume that M_A and M_B use different clients, that is, different versions of the same client or alternative software implementations of the same consensus rule set. If one of these clients happens to have a bug, it may accept a block that is not accepted by other implementations or, conversely, not accept a block that is accepted by other implementations. Both situations may cause a chain split (Kim et al., 2018). Similar situations may arise if the consensus rules are not well specified, i.e. if there is room for interpretation. Client incompatibilities usually get resolved quickly, as there is no incentive to keep using a faulty version of a client once a bug is detected.

A *deliberate* protocol-based fork is what most people think of when they use the term "fork". It emerges when part of the network decides to alter the blockchain's rule set and proceed with an adapted consensus protocol. In many cases, a new distinct cryptocurrency is created from this type of fork. It is therefore rather unsurprising that this type receives most attention and public awareness.



(a) Soft fork (b) Hard fork (c) Forced fork

Figure 3: The block acceptance set of the old (S_{old}) and new (S_{new}) client implementation in the case of a (protocol-based) soft fork, hard fork or forced forks. Process-based forks are excluded since the old and the new blok acceptance sets are equal. Visualization in the style of (Berentsen and Schär, 2017).

To proceed with our analysis of protocol-based forks, we must assume that there is a consensus rule change. For the sake of readability, we rewrite the block acceptance sets A and B as S_{old} and S_{new}. We further assume that any rule changes will be reflected in S_{new}, while S_{old} will stick to the original rules. This notation allows us to propose a formal sub-classification model for protocol-based forks, with the three subtypes: *soft forks*, *hard forks* and *forced forks*. The sub-classification depends on the nature and the implementation of the rule change. As we will see in the next section, the outcome and persistence of protocol-based forks, including the question whether the fork emerges in the first place, is contingent on whether the new block acceptance set S_{new} is a proper subset or a proper superset of the old block acceptance set S_{old}, or whether the two differ significantly in the sense that neither one is a proper subset of the other. Let us first have a look at the three sub-types and then analyze the persistence under different assumptions. The three possible types are shown in Figure 3 and described below.

2.2.1 Soft Fork

A soft fork refers to a change in the consensus protocol, in which the rules for creating new blocks become stricter (Berentsen and Schär, 2017), such that the new block acceptance set S_{new} is a proper subset of the original block acceptance set S_{old}.

$$S_{new} \subset S_{old} \tag{1}$$

As a result, blocks that are valid under the new implementation will always get

accepted by anyone using the old implementation. However, in most cases, the new client will not accept blocks generated by anyone who is applying the old rule set. This becomes apparent when we look at a simple example of a soft fork: a block size limit decrease. Let us assume that some individuals in the Bitcoin network want to decrease a 1 MB block size limit to 512KB (0.5 MB) and thereby halve the number of transactions that can be included in each block. Anyone who is applying the new rule set will produce blocks with a maximum size of half a MB. Since these blocks are below the original 1 MB block size limit, they will get accepted by either party.[4]

2.2.2 Hard Fork

A hard fork refers to the reverse situation. The new implementation makes the consensus rules for block creation less strict (Berentsen and Schär, 2017). Consequently, the new block acceptance set S_{new} is a proper superset of the old block acceptance set S_{old}.

$$S_{new} \supset S_{old} \tag{2}$$

The consequence of this is that blocks created using the new implementation can be rejected by old clients. Conversely, new clients will always consider blocks that are generated in accordance with S_{old} as valid. A block size limit increase is a vivid

[4]A more relevant – and more complex – example is Bitcoin's SegWit soft fork. An in-depth analysis of the SegWit implementation would go far beyond the scope of this paper. In very simple terms, segregated witness introduces a new transaction type with the goal to move signature data to a separate data structure and thereby fixes an issue known as transaction malleability. Nodes who use the old consensus rules are unable to observe the new locking condition and will therefore still accept blocks that have been created under the new rules.

example of a hard fork. Let us assume that some individuals in the Bitcoin network want to increase the 1 MB block size limit to 2 MB and thereby double the number of transactions that can be included in each block. Anyone who is applying the new rule may potentially produce blocks with a size greater than 1MB, that is, greater than the limit of the original consensus rules. These blocks will be rejected by anyone who is applying the old rule set.

2.2.3 Forced Fork

A forced fork corresponds to a change in the rule set that will inevitably lead to incompatibilities and cause a permanent chain split. The new block acceptance set S_{new} is neither a subset nor a superset of the old block acceptance set S_{old}, or more formally:

$$(S_{new} \setminus S_{old} \neq \emptyset) \wedge (S_{old} \setminus S_{new} \neq \emptyset) \tag{3}$$

Where $S_{new} \setminus S_{old}$ denotes the relative complement, such that

$$S_{new} \setminus S_{old} \stackrel{\text{def}}{=} \{x \in S_{new} | x \notin S_{old}\} \tag{4}$$

meaning that both relative complements must not correspond to the empty set.

It refers to a situation in which the old clients will not unconditionally accept blocks created in accordance with the new clients' rule set and vice versa. Eventually, a forced fork with non-zero allocations of consensus-relevant resources for both block acceptance sets will always lead to a chain-split. Good examples include severe

changes in the scripting language, the transaction logic or the signature scheme.

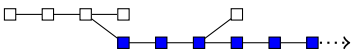
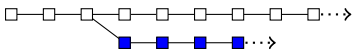
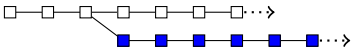
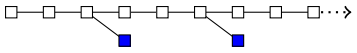
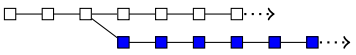
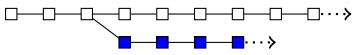
	S_{new} dominant ($r_{new} > r_{old}$)	S_{old} dominant ($r_{new} < r_{old}$)
Soft fork		
Hard fork		
Forced fork		

Table 2: Persistency analysis and ledger development with soft, hard and forced forks, depending on the allocation of consensus-relevant resources. White = blocks generated in accordance with block acceptance set S_{old}. Blue = blocks generated in accordance with block acceptance set S_{new}. Forks occur respectively at $h = 3$. We further assume $r_{new} > 0$ and $r_{old} > 0$. Visualization in the style of (Berentsen and Schär, 2017).

3 Persistency Analysis

The outcome and persistence of process-based forks exclusively depend on the allocation of consensus-relevant resources and the speed of block propagation, which is subject to network topology. The outcome and persistence of protocol-based forks depend on an additional factor: the nature of the protocol change as discussed in our sub-classification in the previous section. We now build on this sub-classification and analyze the different types of protocol-based forks in greater detail. Table 2 visualizes the outcomes of our persistency analysis.

3.1 Forced Fork Persistency

For forced forks, there is always a certain risk that new blocks will only get accepted by a subset of the clients. The likelihood of a permanent chain split, given a change in the block acceptance set that fulfills our criteria of a forced fork, depends on the size of intersection $S_{old} \cap S_{new}$ in relation to the size of S_{old} and S_{new}.

Now let us define R as a measure for the aggregate consensus-relevant resources. We further assume that $R = r_{old} + r_{new}$ where r_{old} and r_{new} represent the consensus-relevant resources employed to find valid blocks in accordance with S_{old} and S_{new} respectively.

Considering the two block acceptance sets as well as the allocation of consensus-relevant resources, we get the probability of a chain split with the creation of the next block.

$$P(b \in S_{new} \vee b \in S_{old}) = \frac{r_{old}}{R}\left(1 - \frac{|S_{new} \cap S_{old}|}{|S_{old}|}\right) + \frac{r_{new}}{R}\left(1 - \frac{|S_{new} \cap S_{old}|}{|S_{new}|}\right) \quad (5)$$

3.2 Soft Fork Persistency

Recall that we classify a fork as a soft fork if it materializes due to a change in the block acceptance set, with $S_{new} \subset S_{old}$.

If the majority of the consensus-relevant resources are allocated to consensus activities in accordance with the block acceptance set S_{new}, meaning $r_{old} < r_{new}$, we would expect the CRN to agree on a generally accepted version of the blockchain. If, on the other hand, the majority of the consensus-relevant resources follow the

old block acceptance set, meaning $r_{old} > r_{new}$, there would be a severe risk of a permanent chain split.

The likelihood of a permanent fork arising with the creation of the next block is given by either of the groups creating a block that will not be accepted by the other group. Given the definition of a soft fork, that any block b created in accordance with S_{new} also is an element of S_{old}, we get

$$P\left[(b \notin S_{old}) \wedge (b \in S_{new} | S_{new} \subset S_{old})\right] = 0. \tag{6}$$

The probability of creating a block that is accepted by S_{old} but is not accepted by S_{new} CRN depends on the relative size of the intersection S_{new}, as well as the relative size of r_{old}.

$$P\left[(b \in S_{old}) \wedge (b \notin S_{new} | S_{new} \subset S_{old})\right] = \frac{r_{old}}{R}\left(1 - \frac{|S_{new}|}{|S_{old}|}\right) \tag{7}$$

This equation corresponds to the soft fork probability that the next block creation event will permanently split the chain.

3.3 Hard Fork Persistency

Recall that we classify a fork as a hard fork if it materializes due to a change in the block acceptance set, with $S_{new} \supset S_{old}$.

If the majority of the consensus-relevant resources are allocated to consensus

activities in accordance with the block acceptance set S_{old}, meaning $r_{old} > r_{new}$, we would expect the CRN to agree on a generally accepted version of the blockchain. If, on the other hand, the majority of the consensus-relevant resources follow the old block acceptance set, meaning $r_{old} < r_{new}$, there would be a severe risk of a permanent chain split.

As in our previous examples with soft forks, the respective probabilities that a fork materializes with the creation of the next block are given by the following two equations.

$$P\left[(b \notin S_{new}) \wedge (b \in S_{old} | S_{new} \supset S_{old})\right] = 0 \tag{8}$$

$$P\left[(b \in S_{new}) \wedge (b \notin S_{old} | S_{new} \supset S_{old})\right] = \frac{r_{new}}{R}\left(1 - \frac{|S_{old}|}{|S_{new}|}\right) \tag{9}$$

This equation corresponds to the hard fork probability that the next block creation event will permanently split the chain.

4 Conclusion

In this paper, we have shown the importance of distinguishing between various types of forks. We have proposed a formal classification framework and analyzed the likelihood and persistency of forks depending on their type and the allocation of consensus-relevant resources. The framework is applicable to a large variety of consensus models. However, some sub-classifications are exclusive to models with probabilistic finality and unrestricted validator pools. In particular, most permissioned

consensus systems may not create process-based forks.

Considering that forks may have severe economic consequences and cause financial damage, the need for a formal classification should be evident. In particular, the framework may help policy makers and practitioners to anticipate potential outcomes and react accordingly. In the following, we discuss some general implications.

First, economic agents must protect themselves from replay attacks, i.e. situations in which a signed transaction is broadcasted on one blockchain network, but then copied and relayed on competing blockchain instances. When a new protocol-based fork emerges, one should not sign and broadcast any transactions until they know that proper replay attack protection is in place. This is of particular importance for organizations that assume the role of a custodian and are responsible for client funds.

Secondly, tokenized assets may be very problematic in the context of forks. Native protocol assets, such as Bitcoin or Ether, are relatively straightforward to handle. They simply split, meaning that the previous owner will receive a version of the asset on each blockchain instance and the market will decide the price of each assets. However, when external promises or real world assets are tokenized, a fork will lead to a situation in which there are multiple tokens but only one physical asset.

Thirdly, forks may raise complicated tax and legal questions, lead to substantial development costs and force economic agents to make decisions under uncertainty. Consequently, each organization should define economic and financial threshold values, as well as other minimum requirements that must be fulfilled for a fork to be considered relevant. Custodians should add these standards to their terms and

conditions and inform their clients that assets assets will only be made available, if they meet the minimum requirements. Our framework helps practitioners and policy makers to take these decisions and to formally define the relevant parameters.

For academics, the framework is a solid foundation for further research and a first step in the direction of a more formal terminology. As such, it will significantly improve our understanding of blockchain forks.

We strongly encourage further research on this topic. Some interesting ideas include the incorporation of velvet forks (Kiayias et al., 2017) and the analysis of the framework in the context of various consensus protocols. Moreover, it would be interesting to collect data on real world forks and see how they can be categorized in our framework.

Acknowledgments

The author would like to thank Tobias Bitterli, Raphael Knechtli, Emma Littlejohn, Jeremias Lenzi, Jakob Roth and Aljoscha Schöpfer as well as the journal editor and reviewers for their valuable inputs.

References

Berentsen, A. and F. Schär (2017). Bitcoin, blockchain und kryptoassets: Eine umfassende einführung. *Aufl. Norderstedt: BoD–Books on Demand*.

Berentsen, A. and F. Schär (2018). a short introduction to the world of cryptocurrencies. *Review of the Federal Reserve Bank of St Louis 100*(1), 1–16.

Eyal, I. and E. G. Sirer (2018). Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM 61*(7), 95–102.

Himmer, K., M. Berchtold, J. Messmer, and P. Sandner (2018). Soft und hard forks: Was sind die wirtschaftlichen und steuerrechtlichen auswirkungen. *FSBC Working Paper*.

Islam, N., M. Mäntymäki, and M. Turunen (2019). Understanding the role of actor heterogeneity in blockchain splits: An actor-network perspective of Bitcoin forks. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*.

Kiayias, A., A. Miller, and D. Zindros (2017). Non-interactive proofs of proof-of-work. *IACR Cryptology ePrint Archive 2017*(963), 1–42.

Kiffer, L., D. Levin, and A. Mislove (2017). Stick a fork in it: Analyzing the Ethereum network partition. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, pp. 94–100.

Kim, S. K., Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey (2018). Measuring Ethereum network peers. In *Proceedings of the Internet Measurement Conference 2018*, IMC '18, New York, NY, USA, pp. 91–104. ACM.

Landoni, M. and G. C. Pieters (2019). Taxing blockchain forks. *SMU Cox School of Business Research Paper* (19-18).

Lin, I.-C. and T.-C. Liao (2017). A Survey of Blockchain Security Issues and Challenges. *IJ Network Security 19*(5), 653–659.

Mehar, M., C. Shier, A. Giambattista, E. Gong, G. Fletcher, R. Sanayhie, H. M. Kim, and M. Laskowski (2017). Understanding a revolutionary and flawed grand experiment in blockchain: The DAO attack. *Journal of Cases on Information Technology* (21), 19–32.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Sapirshtein, A., Y. Sompolinsky, and A. Zohar (2016). Optimal selfish mining strategies in Bitcoin. In *International Conference on Financial Cryptography and Data Security*, pp. 515–532. Springer.

Sompolinsky, Y. and A. Zohar (2015). Secure high-rate transaction processing in Bitcoin. In *International Conference on Financial Cryptography and Data Security*, pp. 507–527. Springer.

Webb, N. (2018). A fork in the blockchain: Income tax and the Bitcoin/Bitcoin Cash hard fork. *North Carolina Journal of Law & Technology 19*(4), 283.

Zamyatin, A., N. Stifter, A. Judmayer, P. Schindler, E. Weippl, and W. J. Knottenbelt (2018). A wild velvet fork appears! inclusive blockchain protocol changes in practice. In *International Conference on Financial Cryptography and Data Security*, pp. 31–42. Springer.

Zhang, R. and B. Preneel (2017). Publish or perish: A backward-compatible defense

against selfish mining in Bitcoin. In *Cryptographers' Track at the RSA Conference*, pp. 277–292. Springer.